CS 241 — Introduction to Problem Solving and Programming

Applied Topics

Towards better data storage:
Multi-dimensional arrays and file I/O

April 20, 2005

Arrays

Arrays are used to store elements of the same type that have an ordering.

Problem:

> Some sets of data have more than one level (or dimension) of order; they make more sense as tables than as lists.

Examples:

> Experimental results based on two variables; matrices; any sort of table. . .

Multi-dimensional arrays

A multi-dimensional array is an object comprised of elements, all the same type, organized and accessible by a series of indices.

Most frequent use: two-dimensional array. Picture as a matrix or table.

	0	1	2	3	4	5
0	1	2	3	4	5	6
1	2	4	6	8	10	12
2	3	6	9	12	15	18
3	4	8	12	16	20	24

Two-dimensional arrays

Declaration/allocation:

```
int[][] table = new int[rows][columns];
```

The size of the table is `rows` × `columns`.

Access:

```
table[i][j]
```

Two-dimensional arrays

Technically, the type of a two-dimensional array is array of array of (base type).

```
table[i][j]
```

Two-dimensional arrays

Technically, the type of a two-dimensional array is array of array of (base type).

```
table[i][j]
```

Two-dimensional arrays

Technically, the type of a two-dimensional array is array of array of (base type).

```
table[i][j]
```

Two dimensional arrays

What is

```
table.length
```

Two dimensional arrays

What is

```
table.length
```

The number of rows

How do you find the number of columns?

Two dimensional arrays

What is

```
table.length
```

The number of rows

How do you find the number of columns?

```
table[0].length
```

Two-dimensional arrays

Addition table example. . .

Multi-dimensional arrays

Since a two-dimensional array is an array of arrays of base types, here's an alternate way of allocating one:

```
int[][] table = new int[rows][]    // an array of int arrays, of length rows

for (int i = 0; i < table.length; i++)
     table[i] = new int[columns];
```

And the rows do not all have to be the same length—nothing is stopping you from making a ragged array.

File I/O

Problem:

Data in computer memory lasts only the duration of the program. Often data needs to saved for a longer-term and retrieved later, either in a later run of the same program or by a different program.

Example: A program you write (the data) in Xemacs must be used later by Xemacs (for revisions) and by javac (for compilation)

Solution:

Write the data to and retrieve it from auxiliary memory (that is, a disk)

File I/O

Input and output is conceptualized by streams.

Java provides a class for writing to a file, `FileOutputStream`:

```
class FileOutputStream {

    public FileOutputStream(String name)

    public void close();
}
```

File I/O

`FileOutputStream`'s methods for writing are very difficult to use (they allow writing only of bytes and byte arrays). To make it easier, there is a `PrintWriter` class which, has an output stream as an instance variable and has more usable methods.

```
class PrintWriter {

     public PrintWriter(OutputStream out);

     public void close();
     public void println(String x);
}
```

File I/O

To read in from a file, Java provides a `FileOutputStream` class. We'll use a class `FileReader`, which automatically generates a `FileOutputStream` as an instance variable.

```
class FileReader {

     public FileReader(String filename);

}
```

File I/O

Finally, the actual reading in of lines of text are done by another class, such as `BufferedReader`.

```
class BufferedReader {

     public BufferedReader(Reader in);

     public void close();
     public String readLine();
     public boolean ready();
}
```